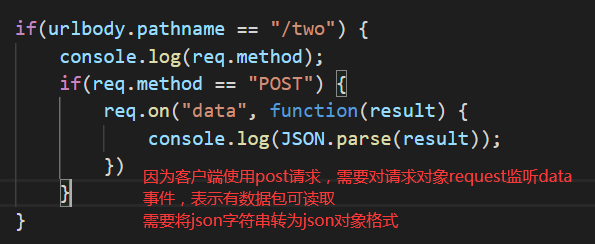
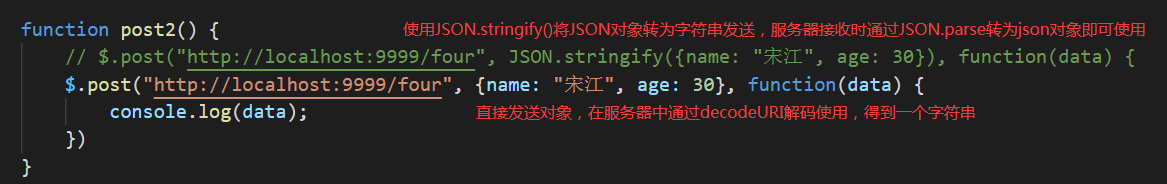
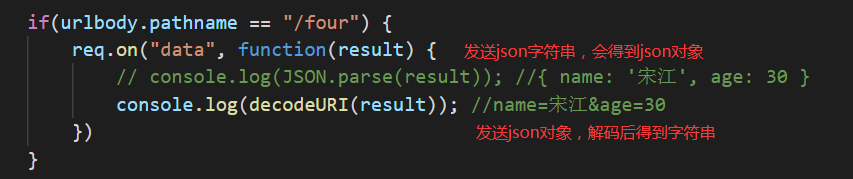
1. 原生ajax请求
   1. 创建ajax对象，并做兼容
   2. 建立连接
   3. 发送数据包
   4. 监听状态改变
   5. 接收响应的数据
   6. setDataToUI





1. jQuery发送ajax请求
   1. $.ajax({method, url, data, success, error})
   2. $.get(url [, data], function(data){})
   3. $.post(url [, data], function(data){})
   4. $.getJSON()





1. Jsonp
   1. 创建一个全局函数等着被调用
   2. 创建一个script标签，并加入到文档树中
   3. 服务器响应时，使用一个函数字符串作为数据响应，函数参数是一个json对象
2. get请求与post请求区别
   1. Get请求是从服务器获取数据，post请求是向服务器传送数据
   2. Get传送的数据量较小，不能大于2KB，post请求传送的数据量较大，一般默认不受控制。但理论上IIS4中最大为80KB，IIS5中最大为100KB
   3. Get安全性非常低，post安全性高，但get执行效率却比post高，建议：
      1. 包含机密信息的话，建议使用post数据提交方式
      2. 在做数据查询时，使用get方式，做数据添加、删除、修改时，建议用post方式
   4. 使用get方式提交，数据会被封装在请求头里，会在网址上显示出来，而post方式提交数据时，数据会被封装在请求体内部。
3. 使用脚手架快速创建以及启动项目
   1. mkdir myegg && cd myegg
   2. npm init egg --type=simple
   3. npm i
   4. npm run dev
4. 项目目录结构
   1. app
      1. router.js 用于配置路由规则
      2. controller 控制层，用于解析用户的输入，调用业务处理层的方法处理后返回相应的结果 C层
      3. serveice 业务处理层，操作数据库 M层
      4. public 用来放置静态资源
   2. config
      1. config.default.js 用于编写配置文件，如修改端口号、mysql
      2. plugin.js 用于配置需要加载的插件
   3. package.json 整个项目的初始化信息，以及项目依赖
5. 服务器基本信息配置

config/config.default.js

config.cluster = {

    listen: {

      path: "",

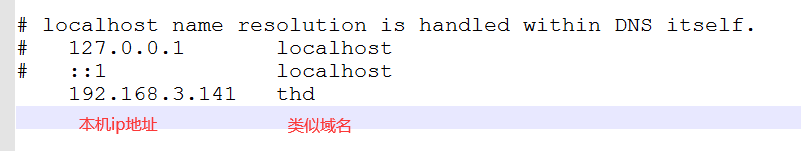
      port: 9999,

      hostname: "thd" //默认是localhost或者ip地址，上线时用0.0.0.0

    }

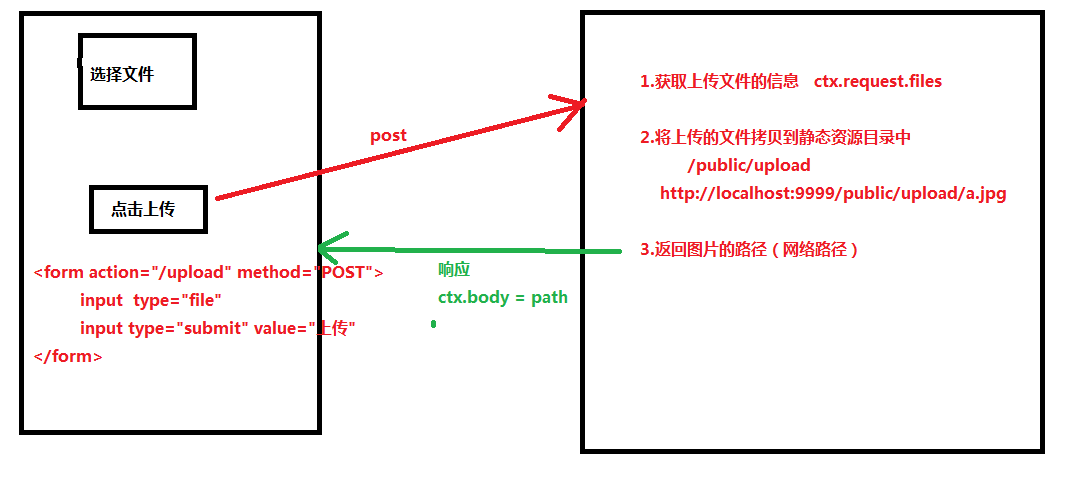
}

C:\Windows\System32\drivers\etc\hosts



请求服务器： http://thd:9999

1. 内置对象
   1. Application，是一个全局对象，在一个应用（服务器）中，是唯一的一个，只实例化一个。在Controller、Service的子类的实例中，可以通过this.app来获取这个全局对象
   2. Request & Response，可以在ctx(context)实例上获取，this.ctx.request/this.ctx.response
   3. Controller, 推荐所有的controller都要继承这个类，有属性
      1. ctx
      2. app
      3. service
   4. Service，推荐所有的service都要继承这个类，属性与Controller一样
2. Path模块
   1. path.dirname()：获取参数中的目录
   2. path.basename()：获取参数中文件名
   3. path.join()：根据多个参数拼接路径
3. 上传文件



1. 跨域处理
   1. 下载插件：

cnpm i egg-cors --save-dev

--save-dev 表示作为当前项目的依赖安装，是下载在项目中的node\_modules中的

* 1. Config/plugin.js

//跨域处理

  cors: {

    enable: true,

    package: 'egg-cors',

  }

* 1. Config/config.default.js

// 跨域配置

  config.cors = {

    origin: '\*',

    allowMethods: 'GET,HEAD,PUT,POST,DELETE,PATCH'

  }

1. 文件上传

// 文件上传

  config.multipart = {

    mode: "file",  //默认是stream，file表示文件

    fileExtensions: [".txt"]

  }

1. Cookies
   1. 可以实现同一个浏览器访问同一个域的时候，不同页面之间的数据数据共享
   2. 实现数据的持久化（关闭浏览器重新打开数据还存在）
   3. 默认情况下，cookies在浏览器关闭以后就销毁了，可以设置有效期来实现数据持久化
   4. 默认情况下，cookies无法设置中文
   5. 设置和获取
      1. this.ctx.cookies.set(key, value, options)
      2. this.ctx.cookies.get(key, options)
   6. 如果要设置中文cookies
      1. 使用加密，encrypt:true
      2. 使用Buffer字符缓冲序列
         1. 存数据：new Buffer(“中文字符”).toString(“base64”)
         2. 取数据：new Buffer(“base64字符串”, “base64”).toString()
   7. 设置有效期
      1. maxAge 毫秒为单位
   8. 签名
      1. signed 默认为true
   9. 加密
      1. encrypt 默认false，不加密
   10. 清除cookie
       1. this.ctx.cookies.set(“username”, null, { maxAge: 0})
2. Session
   1. 是另一种记录客户状态的机制，不同的是cookie保存在客户端浏览器中，而session保存在服务器
   2. 工作流程：当浏览器（客户端）向服务器发起第一次请求时，服务器会创建一个session对象，生成一个类似与key、value的键值对，将这个键值对返回到浏览器（客户端）保存起来。浏览器下次访问时，会携带这个key（cookie），找到对应的session
   3. Session是基于cookies的
3. MySQL
   1. 下载插件：npm i egg-mysql --save-dev
   2. 配置mysql
      1. Plugin.js

// 配置MySQL数据库

  mysql: {

    enable: true,

    package: "egg-mysql"

  }

* + 1. Config.default.js

// 配置数据库

  config.mysql = {

    client: {

      host: "localhost", //主机名、地址

      user: "root", //用户名

      password: "root", //密码

      port: 3306, //端口号

      database: "mydb" //使用的数据库

    }

  }

 CRUD  Create（insert） Read（select） Update（update） Delete（delete）